开发者说系列丛书

C++ STL 程序员开发指南

彭木根 王淑凌 编著

- 用较大的篇幅对C++语言编程核心技术、C++关键库类和 C++STL编程技术进行了深入地剖析和源码解析。
- C++领域内的一本权威著作，是目前市场上惟一采用实例的形式全面讲述STL技术的书籍。
- 融合并提炼了许多程序员多年开发C++程序积累下来的成熟经验。
- 书中的内容、知识点、实例既适合课堂教学，也适合读者自学。

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

C++STL 程序员开发指南

彭木根 王淑凌 编著

中国铁道出版社

2003 · 北京

（京）新登字 063 号

内 容 简 介

本书通过对大量程序实例的分析，深入浅出地讲解了 C++ STL 高级编程技术。全书首先介绍了 C++语言的基本知识、C++语言编程核心技术和 C++关键库类，然后逐步过渡到 C++ STL 编程技术，用较大的篇幅对它们进行了深入的剖析和源码解析。

本书由大量的源程序实例组成，融合并提炼了许多人多年开发 C++程序积累下来的成熟经验，意在展现深澳及抽象的 C++编程技术，特别是令人望而生畏的强大的 STL 技术。

本书是 C++领域内一本权威的著作，是目前市场上惟一一本采用实例的形式全面讲述 STL 技术的书籍。书中的内容、知识点、实例既适合课堂教学，又适合读者自学。无论是高等院校计算机及相关专业的学生，还是计算机业界的从业人员，以及广大的计算机爱好者，都可以从本书中获益。

图书在版编目（CIP）数据

C++ STL 程序员开发指南/彭木根，王淑凌编著. 北京：中国铁道出版社，2003.3
（开发者说）
ISBN 7-113-05164-2
Ⅰ.C… Ⅱ.①彭… ②王… Ⅲ.C 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2003)第 017707 号

书　　名：C++STL 程序员开发指南
作　　者：彭木根　王淑凌
出版发行：中国铁道出版社（100054，北京市宣武区右安门西街 8 号）
策划编辑：严晓舟　郭毅鹏
责任编辑：苏　茜　彭立辉
封面设计：孙天昭
印　　刷：河北省遵化市胶印厂
开　　本：787×1092　1/16　印张：32　字数：754 千
版　　本：2003 年 4 月第 1 版　2003 年 4 月第 1 次印刷
印　　数：1～5000 册
书　　号：ISBN 7-113-05164-2/TP・900
定　　价：56.00 元

目　录

第一篇　预备知识

第二篇 C++ STL 技术原理和组成

第一篇

预备知识

本篇导读

在着手学习和研究 C++ STL 高级编程技术之前，需要您能够熟练地运用 C++编程并分析程序，这些基本的技能通常是从 C++课程以及其他分散的课程中学到的。本书的第一部分旨在帮助你回顾一下这些知识点，包括 C++的一些基本特性以及 C++标准类库。

第 1 章我们将回顾 C++ 的一些特性。因为不是针对 C++ 新手，因此没有介绍诸如赋值语句、if 语句和循环语句（如 for 和 while）等基本结构和基本语法，而是主要介绍一些可能已经被你忽略或忘记的 C++特性，如类的共享成员、保护成员和私有成员；类的继承和多态，友元和纯基类等技术。

第 2 章主要讲述了 C++标准库技术。由于 C++ STL 是 C++ 标准库技术中的一部分，全面了解 C++标准库技术是理解和掌握 C++ STL 技术的基础。本章首先介绍了什么是 C++标准库，然后介绍了输入/输出流，字符串类以及标准异常类等机制。至于标准模板库类技术（C++ STL），将在下面的部分集中介绍。

Chapter 1

C++编程技术

本章主要内容

- C++与 C语言区别
- C++存储技术
- C++函数技术
- C++面向对象机制实现

C++语言是在基于 C 语言的语法基础上融入了 Simula67、Algol68 和 Ada 等语言中许多独特的语法特点之后而形成的全新的语言体系。但由于 C++语言的语法结构与 C 语言的语法结构几乎一致且多数编译器供应商所提供的编译器即可编译 C 语言，同时也可编译 C++语言而且还允许这两种语言混编，因而在很大程度上使许多读者误以为 C++语言是 C 语言的改进型语言。另外在事实上，C++语言也的确可以沿用 C 语言所支持传统的面向过程的分析和设计方法编制出过程化的程序，从而使更多的人对其产生了上面的误解。正是由于这些问题的存在，那些打算自学 C++语言的读者便面临着更大的误入歧途的可能性。至于将 C++语言和 C 语言融入一个编译器来进行编译处理主要还是由于存在商业上需求的缘故。

在所有的 C++语言的最基本的概念中，类（class）便是核心。读者如果已系统地学习过面向对象的分析与程序设计方法，相信学习本章的内容将不会有很大的困难。在掌握了本章论述的 C++语言的最基本的概念之后，再回头重温面向对象的程序设计方法便会更深刻地体会其精深之处了。本章主要介绍 C++的关键语法，分析 C++的许多内核技术。

1-1 C++与 C 语言的区别

C 是一种简单的语言。它真正提供的只有宏、指针、结构、数组和函数。不管什么问题，C 都是依靠宏、指针、结构、数组和函数来解决。而 C++不是这样，宏、指针、结构、数组和函数当然还存在，此外还有私有和保护型成员、函数重载、缺省参数、构造和析构函数、自定义操作符、内联函数、引用、友元、模板、异常、命名空间等。

对每个人来说，习惯 C++需要一些时间，对于已经熟悉 C 的程序员来说，这个过程尤其令人苦恼。因为 C 是 C++的子集，所有的 C 的技术都可以继续使用，但很多用起来又不太合适。正是由于 C++语言的语法是基于 C 语言，并考虑到大多数学习 C++语言的读者应当系统地掌握传统的软件工程所述的面向过程的分析与设计方法和 C 语言程序设计等，并且完全掌握面向对象编程的思想，所以本节是以此为起点来讲述 C++语言的，在同 C 语言进行比较的基础上来论述 C++语言的语法。

1-1-1 文件扩展名的改变

为了使编译器能够区别是 C 语言还是 C++语言，C++语言体系规定用“cpp”（意即 C Plus-Plus）做为 C++语言源文件的扩展名以区别于 C 语言用的“.c”文件扩展名。虽然仅差两个字母，但编译时的处理却相差甚远。后面将不再重复 C 语言已有的内容，所以凡是讲到的多属于 C++语言特有的语法描述的内容则只能书写在以“cpp”为扩展名的文件中（即按 C++语法处理）。反过来按 C 语言语法书写的程序却大多可以使用“cpp”的文件扩展名并用 C++编译器处理。但是也有个别程序易发生一些不符合预期要求的问题（如将在后面讲的重载）。所以为了避免混乱，建议还是按各自语法规则书写。“cpp”的文件扩展名与操作系统无关。与 C++语言源文件相关的头文件扩展名一般仍用“h”，但有些操作系统也有规定使用“hpp”充当头文件扩展名的。

1-1-2 简化输入/输出手段

在C语言中，输入/输出本是依靠函数来实现的（如标准输入/输出用的scanf、printf等）。通常只要在程序的最前端放置对应的头文件声明“#include<stdio.h>”就可以引用这类函数。由于此种函数的引用语句书写起来比较冗长，为了简化起见C++语言又另外定义保留字和运算符来替代C语言中对标准输入/输出函数的引用。C++语言的保留字为：

cout<<"输出内容"<<…； /*为标准输出算符（默认为显示器）*/

cin>>"输入内容">>…； /*为标准输入算符（默认为键盘）*/

支持此二算符的内部函数体在一个名为“iostream.h”（即标准I/O流）的头文件中予以声明，所以应用时一定要将其放置对应的头文件声明部位。

【例程1-1 简化输入/输出手段实例】

```
//文件名: CHAPTER2-1.cpp
#include<iostream.h>
void main()
{
char  name[10];
   int  age;
   cout<<"please  input  your  name:";
   cin>>name;
   cout<<"How  old  are  you:";
   cin>>age;
   cout<<"name  is  "<<name<<"\n";
   cout<<"age  is  "<<age<<"\n";
}
```

将实例中的标准输入/输出语法与C语言的printf和scanf函数的书写语法对比后可以很明显地看出此种标准的输入、输出语法的书写大大简化了函数格式的描述，而由C++语言编译器按被操作数的类型和具体内容代为选定默认格式。实际上此种方法也有其一整套控制格式的手段。运行时在屏幕输出如下：

```
please  input  your  name: peng
How  old  are  you: 20
name  is  peng
age  is  20
```

1-1-3 数据类型声明的改变

C++语言除了新增的类数据（即class）类型以外，还继承了C语言所支持的所有数据类型。但在数据类型的声明上作了两种较大的改变，一是允许数据类型的声明语句可以出现在程序的任何位置；二是允许直接使用结构体名定义实体。

【例程1-2 数据类型声明实例】

```
//文件名: CHAPTER2-2.cpp
#include <iostream.h>
void  main()
```

```
{
   struct  Dt{unsigned  int  mm,dd,yy;};
   Dt dt[3];
   for(int i=0;i<3;i++) {
     dt[i].mm=8;  dt[i].yy=1994;  dt[i].dd=i+15;
       cout<<"mm: yy: dd "<<dt[i].mm<<" "<<dt[i].yy<<" "<<dt[i].dd<<"
"<<endl;
   }
}
```

运行时在屏幕输出如下：

```
mm: yy: dd 8 1994 15
mm: yy: dd 8 1994 16
mm: yy: dd 8 1994 17
```

在 C 语言中用“/*…*/”符号做程序注释，这种注释叫做段注释。当只做单行注释时便可用连续的两个“//”符号表示从此符号起到该行的末尾均为行注释内容。这种注释原在 C 语言中就有效，但一直未公开发表。

在 C++语言编程设计中常利用#define 语句定义一类空的字符串代换技术被称为代名符定义。代名符在程序中不起任何实际作用，但只能放在行首用来说明程序的设计者、类的归属、设计时间、程序的分类或所处的总体位置等，有助于读程序的信息，如下所示：

```
#define  TEST
TEST  void  main()
{…}
```

1-1-4 动态内存分配运算符的使用

C++与 C 语言相比较，除了上面的改进外，还可以用来进行动态内存分配。这种运算符有两个：new 和 delete。这是一对真正的运算符，无须任何函数支持（即由 C++语言编译器直接处理）。C++语言之所以要以 new 和 delete 运算符来代替 C 语言内繁多的内存动态分配函数，是由于在面向对象的程序运行中，对象的产生和撤消极其频繁，以至于用 C 语言中动态内存分配函数来完成这种内存的占用和释放不仅过于烦琐，甚至是不可能的。

New 运算符的功能等效于 C 语言中 malloc 一类的函数功能，被用来动态地为对象或数据分配内存。即在尚未占用的内存空间中为对应类型的数据的实际需要来安排空间，并带回分配的首址。其语法格式为：

指向对应类型的指针=new **类型描述符**;

其中的类型描述符可有下面几种书写形式：

- 仅占一个单元空间：

```
int  *p;
  p=new  int;        /*意即占一个机器字长*/
```

- 仅占一个单元空间且赋初值：

```
int   *p;
  p=new  int(20);  /*意即占一个机器字长且赋予初值 20*/
```